IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 9 June 2024, accepted 24 June 2024, date of publication 3 July 2024, date of current version 16 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3422261

RESEARCH ARTICLE

Enhanced Branch and Bound Algorithm: Minimizing Subproblem Complexity in Power Dispatch

ELIF CESUR[1], MUHAMMET RAŞIT CESUR[1], AND AJITH ABRAHAM[2], (Senior Member, IEEE)
[1]Industrial Engineering Department, Istanbul Medeniyet University, 34700 İstanbul, Türkiye
[2]School of Computer Science Engineering and Technology, Bennett University, Greater Noida, Uttar Pradesh 201310, India
Corresponding author: Elif Cesur (elif.karakaya@medeniyet.edu.tr)

ABSTRACT The Branch and Bound (BB) algorithm, while ensuring optimality, often encounters performance bottlenecks, characterized by slow execution and high computational overhead, especially when dealing with intricate or extensive problem instances (NP-Hard). This study introduces an innovative approach by dividing the problem into partial (local) problems in a manner that would not compromise optimality and solving sub-problems of each local problem individually, to shrink the solution space. In the initial phase, this research establishes and validates the mathematical foundation of the proposed algorithm, which involves a pruning approach. Subsequently, enhancements are incorporated into the existing BB to partition the solution space into more manageable sub-spaces and consolidate solutions from these sub-spaces. In the final phase, the Enhanced Branch and Bound (EBB) algorithm is applied to a real-world power dispatching optimization case study. The outcomes of this investigation reveal the following: 1) For smaller problem instances, both the conventional BB and the proposed EBB algorithm yield identical optimal solutions. 2) In contrast, the EBB algorithm demonstrates significantly improved performance in solving NP-hard problems that pose challenges for the BB and BB with pruning. The primary contribution of this research is the introduction of EBB, an enhanced version of BB, specifically designed to effectively tackle NP-hard problems. This approach can be integrated with all pruning, branching, and bounding strategies used in BB, thereby boosting its performance and making it applicable to all problems solved by BB variations.

INDEX TERMS Branch and bound algorithm, pruning strategy, enhanced branch and bound algorithm, power dispatching optimization.

I. INTRODUCTION

The Branch and Bound (BB) algorithm arises from the need to efficiently solve complex optimization problems involving multiple constraints and objectives. It provides a systematic approach to exploring large solution spaces, making it invaluable when exhaustive searches are impractical. It is also a reliable tool for obtaining optimal solutions or proving optimality in scenarios where precision is crucial. The algorithm's mechanism involves the creation of subproblems, referred to as the ''branching'' phase, while the subsequent ''bounding'' phase is dedicated to the elimination of subproblems that do not contribute to the solution. The BB involves systematically exploring all potential solutions for the given problem by retaining and managing partial solutions referred to as ''subproblems'' within a structured tree framework.

While the BB provides powerful solutions to optimization problems, it faces critical challenges such as addressing exponential growth in problem size and minimizing computational time. Current research is primarily focused on enhancing the efficiency of the BB by incorporating an innovative approach. This approach include optimizing branching strategies and pruning techniques to remove redundant nodes, as well as utilizing parallelization to improve processing speed. A notable

The associate editor coordinating the review of this manuscript and approving it for publication was Vitor Monteiro.

research challenge remains: How can further development and improvement of the BB be undertaken to more effectively address these challenges, ensuring its applicability to larger and more complex optimization problems while ensuring optimality?

In this study, three versions of BB algorithm have been utilized: 1) BB denotes the conventional branch and bound algorithm, elucidated both mathematically and algorithmically, 2) BB with pruning (BBP) refers to one of the existing pruning strategies from literature, and 3) An Enhanced Branch and Bound (EBB) algorithm is proposed to boost the problem solving speed of BB, which has been incorporated into the text to demonstrate and substantiate the evident performance of EBB.

Our motivation is to eliminate the sub-problems generated by BB that do not lead to optimal solutions, thereby reducing the overall problem-solving time. This approach enhances ability of BB to tackle NP-Hard problems efficiently. We achieve this by dividing the main problem into independent partial problems.When the solutions of the partial problems are merged, the optimal solution will result. We conducted a mathematical investigation about how to divide the problem into partial problems without impacting the optimal solution, coining this approach as EBB. The proposed EBB can be applied to pruning, branching strategies, and bounding strategies similar to BB, making it an enhanced version of the BB. Finally, we have chosen to address an energy manufacturing problem due to the critical significance of energy resources in today's world. By focusing on this sector, we aim to highlight the real-world applicability and relevance of our proposed methodology, while also emphasizing the broader importance of energy-related optimization in various industrial contexts

This research's primary contribution lies in the efficient delivery of rapid and optimal solutions for NP-hard problems by the EBB. It achieves this by dynamically shrinking the solution space during runtime, thereby effectively reducing the number of iterations required.

II. LITERATURE REVİEW

NP-hard problems represent a formidable obstacle in the realm of optimization studies, primarily due to their inherent complexity. The task of solving NP-hard problems is notably demanding. To address this challenge, advanced heuristic techniques have gained widespread acceptance. These methods aim to provide near-optimal solutions swiftly [1]. Nevertheless, the quest for achieving optimality has led to the development of quantum optimization techniques [2], [3] alongside the refinement of algorithms designed to pinpoint the optimal solution. The BB serves as a valuable tool for obtaining optimal solutions to mathematical and computational problems that pose challenges to efficient polynomial-time solutions. Several surveys exist that compile research related to the BB, providing a historical perspective on its evolution. Tomazella and Ngano offer a comprehensive review of the BB specifically applied to flow-shop scheduling problems, delving into the ''method's evolution and its various components [4]. Morrison al. survey recent advancements in searching, branching, and pruning strategies within the BB framework [5]. Additionally, Huang et. al. survey different techniques and emerging trends in the BB for solving mixed-integer linear programming problems, including the incorporation of machine learning to enhance algorithmic components [6].

A literature review of the BB reveals various approaches and modifications aimed at enhancing its efficiency and effectiveness in solving optimization problems. One such approach that has been explored involves the utilization of the recursive BB [7]. This approach entails recursively subdividing the problem space and assessing the suitability of further processing for each segment [8]. Other strategies in the literature encompass the application of diverse heuristics and bounding techniques to improve the performance of the BB. For instance, depth-first search and best-first search techniques can be employed to enhance the efficiency of exploring the set of candidate solutions. Kariwala and Cao present an upgraded BB that employs a best-first search approach, which has demonstrated superior efficiency compared to depth-first search approaches [9]. Another research area revolves around parallelizing the BB. This parallelization allows for the concurrent exploration of different branches and can result in substantial speed-ups in solving optimization problems. Pruul, Nemhauser, and Rushmeier discuss the pioneering use of parallel processing in a BB for addressing the traveling salesman problem [10]. Additionally, Laursen introduces two parallel BBs, one featuring a static distribution of subproblems and limited communication, and the other incorporating dynamic distribution and synchronized communication [11]. Both of these approaches exhibit high processor utilization.

Over the years, it has evolved and has been extended to address a wide range of optimization problems, both discrete and continuous. Mersha and Dempe propose an extended BB for linear bilevel programming problems, enabling the inclusion of arbitrary linear constraint functions at the upper level [12]. Gao, Mishra, and Shi further extend the algorithm to solve the sum-of-nonlinear-ratios problem. Rendl, Rinaldi, and Wiegele present a BB for the Max-Cut problem, which combines semidefinite and polyhedral relaxations [13]. Additionally, Przybylski and Gandibleux highlight the introduction of the multi-objective BB, which expands the method's capability to handle multiple objectives [14].

Several papers present various applications and enhancements of the BB in diverse optimization problems. Woodcock and Wilson introduce a hybrid algorithm that combines the BB with tabu search to tackle the generalized assignment problem [15]. In contrast, Zabudsky and Veremchuk delve into the application of the BB for solving the Weber problem involving rectangular facilities on lines, even in the presence of forbidden gaps [16]. When it comes to locating every

Pareto solution within a biobjective mixed integer algorithm, Adelgren and Gupte provide a BB [17]. Furthermore, Elaibi discusses the utilization of the BB in determining the nearest link during the construction of a railway network [18]. Lastly, Alvarez-Valdes, Martinez, and Tamarit specifically address cutting and packing problems involving irregularly shaped pieces, presenting an exact BB that exhibits commendable performance [19].

Power dispatching refers to the process of optimally distributing available power resources to meet the demand in an electrical network. The BB can be employed to optimize power dispatching by determining the most efficient allocation of resources and minimizing transmission losses, thereby ensuring a reliable and stable power supply. To address the issue of reactive power dispatch and planning in electrical power networks, Estevam et. al. suggest a non-linear BB [20]. Lendak, Vidacs, and Erdeljan present an algorithm for automatic one-line diagram generation for power distribution systems, leveraging the BB [21]. Furthermore, Constante, Lopez, and Rider focus on optimal reactive power dispatch with discrete controllers and utilize a BB in conjunction with semidefinite relaxation [22]. Finally, Frag, Baiyat, and Cheng discuss the optimization problem concerning real and reactive power and introduce a novel algorithm for load shedding and reallocation of generation using linear programming techniques [23].

While the literature reveals a diverse range of approaches for addressing optimization problems through BBs, such as Lagrangian relaxations, linear programming, dynamic programming, integer programming, constraint programming, and artificial intelligence, there remains a notable research gap. Specifically, current research primarily focuses on enhancing the efficiency of the BB by comparing lower and upper bounds to eliminate redundant nodes. In contrast, our proposed approach seeks to address this gap by dividing the problem into sub-problems and generating solutions within each sub-problem. This strategy aims to reduce the overall number of search iterations, ultimately shrinking the solution space a novel pathway that has yet to be extensively explored in the existing literature.

III. THE APPROACH

The BB represents an approach to discovering optimal solutions. Nevertheless, it is important to note that NP-Hard problems cannot be efficiently solved by BB due to their time complexity. To enhance the algorithm and facilitate the solution-finding process, various strategies like pruning have been proposed. In this regard, we present an innovative approach aimed at reducing the size of solution space by partitioning the problem into independent local problems.

A. OVERVIEW OF THE BRANCH AND BOUND

In numerous optimization problem domains, the BB has demonstrated its effectiveness in providing precise solutions. The BB employs a tree-based structure to systematically generate and evaluate all possible solutions for a given problem. It applies pruning to remove branches within the search space that do not contribute to an improved solution. Within the BB framework, three key components stand out as potential areas for modification, offering solutions to enhance the algorithm's overall performance [5]:

Search Strategy: The search strategy dictates the order in which the algorithm explores various facets of the problem space. It can be likened to the process of choosing which paths to traverse first in a complex maze, to efficiently navigate through the labyrinth of possibilities to reach the optimal solution.

Branching Strategy: The branching strategy outlines the approach the algorithm takes to decompose the original problem into smaller, more manageable segments. This concept is similar to breaking down a challenging task into a series of more tractable subtasks, thereby simplifying the overall problem-solving process.

Pruning Rules: Pruning rules serve as guiding principles that assist the algorithm in avoiding unproductive regions within the problem space. They function similarly to eliminating dead-end pathways in a maze, allowing the algorithm to disregard paths that do not hold promise for improving the solution quality.

Before delving into a detailed explanation of algorithms for the BB and the proposed EBB in detail, it is essential to establish the notations that will be used throughout the discussion. D represents the search domain, which is a collection of all possible solutions to the problem. $f\ (n) : D \rightarrow \mathbb{R}$ is the objective function that assigns a real number to each solution in the search space. It starts by building a search tree, denoted as T, consisting of subproblems of the search space. The algorithm selects a new subproblem, represented as S from a list of unexplored subproblems, symbolized as L. Bounding procedures involve calculations performed at each node N in the search tree. Lower Bound (LB_n) is a value computed at each node, representing the lowest possible value (bound) for the corresponding subproblem. It's like finding the minimum cost or value achievable within that particular branch of the tree. Upper Bound (UB_n) involves finding a feasible solution with an upper bound for the global minimum. In simpler terms, it's like identifying a solution that is not worse than the best solution found so far. The goal is to find the best solution, denoted as x^*, which maximizes the value of the objective function, $x^* \in \arg\min x \in Df(n)$.

If the algorithm finds a solution within this subproblem, symbolized as x_c^* with a better objective value than the current solution in other words $f\ (x_c) < f(x^*)$, it updates the solution to be x^*. Then, the algorithm generates child subproblems by dividing S into a set of subproblems represented as S_i subproblem is then added to the search tree T.

If it can be proven that no solution within the subproblem S_i has a better objective value than the current solution means that $f\ (x_c) > f(x^*)$, the subproblem is pruned.

The algorithm continues this process until there are no unexplored subproblems left. At that point, it returns the

best solution found. Since subproblems are only discarded if they contain no solutions better than the x^*, it is guaranteed that the returned solution x^*, is the optimal solution that maximizes the objective function within the entire search D. The pseudocode for the basic BB is given below.

Algorithm 1 The BB for Maximization Problem

```
1.  LB* ← −∞, UB* ← ∞, x* ← −∞
2.  Set L = {D} and initialize, x*
3.  while L ≠ ∅
4.      Set a Sub-problem S_i ← {L}
5.      Select a Node N from S_i
6.      Generate neighborhood for N_i
7.      Insert all neighbors into S_i
8.      for all Sub-node S_i do
9.          Calculate UB_n of N_i and set x_n^UB ← UB_n
10.         if UB_n > UB*
11.         Calculate LB_n of N_i and set x_n^LB ← LB_n
12.         if LB_n > LB*
13.             Set x* ← x_n^LB and LB* ← LB_n
14.         end if
15.         if Sub-problem S_i cannot pruned
16.             Divide S_i into Sub-problems
17.         else if
18.             Remove S_i from L
19.         end if
20.     end for
21.   end while
22. return LB*, UB*, x*
```

B. ANALYTICAL PROOF OF THE ENHANCED BRANCH AND BOUND'S CAPABILITY

In the BB, at the initial state where r represents the root node, the upper bound (UB_r) is the sum of all possible gains, while the objective function value, the lower bound (LB_r) is 0. Expressing the sum of potential gains as the function $f(r)$, equation (1) illustrates the relationship between the upper bound and lower bound.

$$UB_r = LB_r + f(r) \tag{1}$$

During the search process, it is expected that the difference between UB_r and LB_r decreases, which means that the value of the function $f(r)$ becomes smaller. When the value of the function $f(r)$ reaches 0 means that $UB_r = LB_r$ it signifies that an optimal solution has been found.

Therefore, it is expected that $f'(r) \leq 0$. It is also expected that the function $f(l)$ at each level (l) of the search tree satisfies $f(l) \geq f(l+1)$. Similarly, $LB_N^l \leq LB_N^{l+1}$ is expected. In contrast, if the lower bound value at lower levels of the tree is smaller than the best lower bound value at higher levels ($LB_N^l > LB_N^{l+1}$), consequently, the upper bound values at the node's higher levels will also be smaller ($UB_N^l > UB_N^{l+1}$).

In this case, the BB will work like brute force to explore lower levels of the tree. For the brute force mode to terminate, $LB_N^l > LB_N^{l+1}$ must end, which means the objective function value needs to start changing positively. When there is a positive change, the best possible result at that point will emerge locally. This is because all nodes will have been evaluated in such a way that there will be no upper bound value above the one that is certain to have decreased. The remaining nodes (if any) will not be evaluated because their upper bound values are smaller.

When the objective function value starts to increase, the BB will continue to work as expected. However, if the objective function value decreases again, it will start a brute-force search for solutions at lower levels of the tree. Let (m) be the level where the first consecutive decrease in the objective function occurs, and (n) be the level where the second consecutive decrease is observed. In this case, because $UB_N^m < UB_N^n$, when level (n) is reached, the solution for the part up to level (m) will no longer change. Therefore, the solution can be obtained by starting from level (m) and solving the problem from We are currently investigating two hypothesesm there, considering only the part up to level $(m-1)$. We investigate two hypothesis at this point.

Hypothesis 1: When two local optima are found at different levels of the tree, the best point at the upper level (with certain optimality) will persist as part of the optimal solution.

Hypothesis 2: Dividing the problem from a point of certain optimality will reduce the total iterations of the BB algorithm.

Proof 1: When the first local optimum is passed at level $m+1$, $LB_m > LB_{m+1}$ is observed for all best nodes in maximization problems, while $UB_m < UB_{m+1}$ is seen for all best nodes in minimization problems at levels m and $m+1$. This occurs because LB_m or UB_m at level m represents a saddle point. There is a possibility that multiple nodes at level m represent saddle points with LB_m or UB_m. Thus, it is necessary to continue iterations to another saddle point at level n to identify the nodes at level m that are connected to nodes representing LB_n or UB_n. After identifying the connected nodes at level m, the problem can be divided, and the remaining portion can be solved by using these connected nodes as the starting points.

Proof 2: The number of nodes contained in the tree up to level m is $2^m - 1$, and the number of nodes contained in the tree up to the final level (l) is $2^l - 1$. When the problem is divided into two parts starting from level (m), the total number of nodes will be as given in equation (2) as T_N.

$$T_N = 2^m - 1 + 2^{l-m} - 1 = 2^m + 2^{l-m} - 2 \tag{2}$$

When comparing the total number of nodes between the problem with two parts and the problem with a single part, equation (3) will be obtained.

$$2^m + 2^{l-m} - 2 <^? 2^l - 1 \Longrightarrow 2^m + 2^{l-m} - 1 <^? 2^l \tag{3}$$

When the term 2^{l-m} on the left side of Equation 3 is moved to the right side and both sides are divided by $2^m - 1$, equation 4 is obtained. Since $l > m$, $l - m > 0$, and $2^{l-m} > 1$. Therefore, when solving a problem with the BB, if the objective function value decreases first and increases later from two distinct points, the solution for the part up to the first point is obtained, and the remaining part is solved again. This allows the algorithm to search for an optimal result in a smaller

solution space.

$$\begin{aligned} 2^m - 1 &<^? 2^m 2^{l-m} - 2^{l-m} \\ \Longrightarrow 2^m - 1 &<^? (2^m 1)2^{l-m} \\ \Longrightarrow 1 &< 2^{l-m} \end{aligned} \tag{4}$$

The proposition that has been proven remains valid when there are more than two points of change (local optima) in the objective function. Therefore, when the problem is divided into segments at local optima where there are consecutive decreases and increases in the objective function, the same principle applies. As a result, the problem can be solved by dividing it into as many segments as there are consecutive local optima, effectively shrinking the solution space each time.

Here's a breakdown of the algorithm's main steps for solving a maximization problem:

Initialization: LB^* is initialized to negative infinity, indicating that there is no lower bound yet. UB^* is initialized to positive infinity, indicating that there is no upper bound yet. x^* (the best solution found so far) and x_p^* (the previous best solution) are initialized to negative infinity. The counter is initialized to 0, used to keep track of the objective function's fluctuation. L is set to a set containing D, which suggests that D represents the search space.

Loop: The algorithm enters a while loop that continues until the set L is not empty. A sub-problem S_i is created, and initialized with L. A node N is selected from S_i. Neighborhoods of N are generated, and all neighbors are inserted into S_i. For each sub-node in S_i, the following steps are executed:

Calculate the upper bound UB_n for the sub-node N_i, and set x_n^{UB} to this value, If UB_n is greater than the current best upper bound UB^*. Calculate the lower bound LB_n for N_i, and set x_n^{LB} to this value, If LB_n is greater than the current best lower bound LB^*. Update x^* to x_n^{LB} and LB^* to LB_n. Check if the sub-problem S_i can be pruned. If it cannot be pruned, divide it into sub-problems. If it can be pruned, remove S_i from L.

Updating Best Solution: After processing all sub-nodes in S_i, the algorithm checks whether x_p^* is greater than the current best solution x^*. If it is, a counter is incremented. If the counter reaches 2, this section of the algorithm divides the problem into two parts from the first fluctuation point. The new root of the second part is assigned as x^* at the first fluctuated level. The bound variable of x^* consisting of lower bound (T_{lb}) and upper bound (T_{ub}) are set as global bound variables (T_x), which are represented by T_n, inserted into L. Otherwise, T_n is inserted into the remaining nodes of S_i, and bounds are updated. The algorithm assigns x_p^* to x^* before the next iteration. The algorithm continues these iterations until L becomes empty. Once the loop terminates, it returns the best lower-bound LB^*, the best upper bound UB^*, and the best solution x^*.

In summary, this algorithm systematically explores potential solutions while keeping track of the best solution found so far, pruning sub-problems that do not hold promise, and efficiently navigating through the solution space to find the optimal solution to a maximization problem. The pseudocode outlines the steps of the EBB given below.

Algorithm 2 The EBB for Maximization Problem

```
LB* ← −∞, UB* ← ∞, x* ← −∞, x_p* ← −∞,
counter ←0
Set L = {D} and initialize, x*
while L ≠ ∅
   Set a Sub-problem S_i ← {L}
   Select a Node N from S_i
   Generate neighborhood for N_i
   Insert all neighbors into S_i
   for all Sub-node S_i do
         Calculate UB_n of N_i and set x_n^UB ← UB_n
      if UB_n > UB*
      Calculate LB_n of N_i and set x_n^LB ← LB_n
      if LB_n > LB*
         Set x* ← x_n^LB and LB* ← LB_n
      end if
      if Sub-problem S_i cannot pruned
         Divide S_i into Sub-problems
      else if
         Remove S_i from L
   end if
 end for
 if x_p* > x*
   counter++
   if counter = 2
     counter ←0
     LB* ← T_lb, UB* ← T_ub, x* ← T_x
     Insert {T_n} to L
 else
           Insert T_n to remaining nodes of S_i
           Set T_lb ← LB*, T_ub ← UB*, T_x ← x*
      end if
   end if
   Assign x_p* asx*
 end while
 return LB*, UB*, x*
```

IV. CASE STUDY

The EBB exploits trade-offs, leading to fluctuations in the objective function's value, to reduce the solution space and achieve faster optimal solution discovery. Among various optimization problems that inherently entail trade-offs, the energy dispatching problem stands out as an exemplary case. It effectively illustrates these trade-offs due to the fluctuation of hourly energy prices throughout each day, resulting in some periods where the price falls below the production cost. Consequently, the EBB was applied to the energy dispatching optimization problem using the Ontario Energy Demand dataset [24].

The dataset includes the hourly energy consumption and unit energy price for the city of Ontario between the years 2003 and 2023. As seen in Figure 1, the energy price fluctuates. Considering both variable and fixed costs of energy production, there are specific time intervals during which energy production results in a loss. To avoid losses,

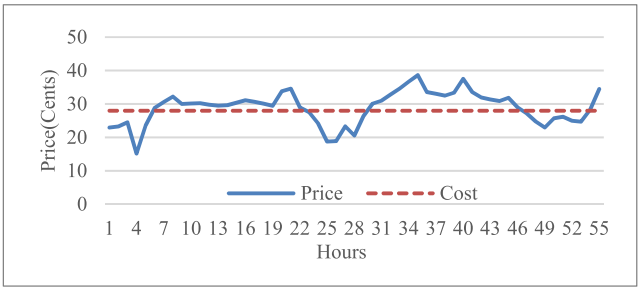

FIGURE 1. The fluctuation of hourly energy prices (cent).

the production can be stopped, but the facility can be restarted within at least 5 hours. In the first hour of operation (Ramp-up state), the facility can generate electricity up to a maximum of 25% of its capacity. After starting operation, the facility must operate for a minimum of 8 hours. During production, the facility can operate in either full capacity (Base Load State) or minimum capacity (Min Load State) mode. In full capacity mode, the entire capacity can be used for production, while in minimum capacity mode, only half of the capacity can be utilized. During the hour when production is halted (Ramp-Down State), electricity production can be up to a maximum of 12.5% of the production capacity.

A. MATHEMATICAL MODEL OF POWER DISPATCHING

The scope of the power dispatching optimization problem is to schedule energy manufacturing activity regarding the dynamic energy price ($\boldsymbol{P_t}$) that changes hourly, aiming to eliminate losses, and maximize profits. The dispatching problem consists of hourly production amounts ($\boldsymbol{A_t}$), capacities ($\boldsymbol{Ca_t}$), setup costs ($\boldsymbol{C_s}$), variable costs ($\boldsymbol{C_v}$), labor costs $\boldsymbol{C_l}$, demand ($\boldsymbol{D_t}$) and prices ($\boldsymbol{P_t}$). Setup costs and labor costs must be considered hourly expenses because of the hourly nature of demand and price data. In that context, the index "t" represents the hour within the model. Labor cost is multiplied by (1 - $\boldsymbol{S_t}$), where $\boldsymbol{S_t}$ is a binary variable representing the facility's operational status. This adjustment accounts for the increased labor costs incurred while the facility is in operation. The objective of our optimization problem is to maximize profit, as defined in equation 5.

$$Z_{max} = \sum_{t=1}^{T} A_t\,(P_t - C_v) - C_s - C_l\,(1 - S_t) \tag{5}$$

The hourly production amount is determined as the minimum value between the hourly demand and the hourly capacity, as described in equations 6 and 7. In the context of power production, the hourly capacity varies depending on the state of the manufacturing facility. When the facility commences manufacturing in the ramp-up state (Ru_t), it produces 25% of its actual capacity. In the minimum load state (M_t), 50% of the actual capacity is produced. The base load state (B_t) corresponds to operating at full capacity. Conversely, when manufacturing is halted in the ramp-down state (Rd_t) production is reduced to 12.5% of the actual capacity.

$$\forall_{t=1}^{T} A_t \le D_t \tag{6}$$

$$\forall_{t=1}^{T} A_t \le Ca_t\,(0.25Ru_t + 0.125Rd_t + 0.5M_t + B_t) \tag{7}$$

The model must select only one state at a time, as presented in Equation 8. Before commencing manufacturing, a minimum of 5 consecutive stopping hours (S_t) is required, as specified in equations 9 and 10. Once production begins, the facility must remain active for a minimum of 8 hours. This includes one hour for ramping up and another hour for ramping down, leaving at least 6 hours for operating in either the minimum load or base load state, as described in equation 11. Additionally, the ramp-down state must follow either the base load or minimum load state for one hour, as stated in Equation 12. Following the ramp-down, the facility must come to a complete stop, as outlined in equations 13 and 14. Finally, binary and non-negativity constraints are specified in equations 15 and 16.

$$\forall_{t=1}^{T} \mathrm{S_t} + \mathrm{Ru_t} + \mathrm{Rd_t} + \mathrm{M_t} + \mathrm{B_t} = 1 \tag{8}$$

$$\forall_{t=5}^{T} \mathrm{Ru_t} = \mathrm{Ru_t} \prod_{i=1}^{5} \mathrm{S_{t-i}} \tag{9}$$

$$\forall_{t=1}^{T-5} \mathrm{S_t} = \sum_{i=1}^{5} \mathrm{Ru_{t+i}} \tag{10}$$

$$\forall_{t=1}^{T-6} Ru_t = Ru_t \prod_{i=1}^{6} (M_{t+i} + B_{t+i}) \tag{11}$$

$$\forall_{t=2}^{T} Rd_t = Rd_t\,(M_{t-1} + B_{t-1}) \tag{12}$$

$$\forall_{t=1}^{T-1} Rd_t = Rd_t S_{t+1} \tag{13}$$

$$\forall_{t=1}^{T-1} S_{t+1} = (Rd_t + S_t) S_{t+1} \tag{14}$$

$$S_t, Ru_t, Rd_t, M_t, B_t \in \{0, 1\} \tag{15}$$

$$A_t \ge 0 \tag{16}$$

B. COMPUTATIONAL EXPERIMENT

The problem was attempted to be solved on a computer with an 11th Gen Intel(R) Core (TM) i7-1165G7 @ 2.80GHz processor, 16 GB of RAM and SSD disc. This problem was solved using the BB, the BBP, the EBB, and IBM - ILOG's Optimization Programming Language (OPL) solver. Assumptions of the model are $10.000,00 hourly setup cost, $40.000,00 hourly labor cost, $25 hourly operation (variable) cost, and 16 MWh production capacity. Since it is an NP-Hard problem, a solution covering the entire dataset could not be optimized with the EBB, OPL, BBP, and BB. However, EBB achieved an optimal solution for problems that BB, BBP, and OPL can not solve within an appropriate time limit.

We compared EBB with BB, BBP, and OPL. The OPL algorithm is widely employed in prevalent commercial software today. The BBP, identified as the most effective pruning strategy by Goerikg et al. [25], is utilized by Coniglio et al. [26]. Given the similarity of our symmetric problem to the one addressed by Goerikg et al., we adopt the same pruning strategy as BBP. Additionally, Segundo et al. [27] observe that this strategy ensures the bound remains unchanged but has a limited impact on reducing branching. 24 hours, 48 hours, 65 hours, 96 hours, and 120 hours scenarios were successfully solved by all methods. The OPL Solver could not find the optimal solution for 96 hours and 120 hours. Also, the OPL solver, the BB, and the

BBP were unable to find a solution within a 10-hour operating window for the 240-hour problem. The EBB solved the 240-hour and 710-day instances. However, it could not solve more than 710 days within the 8-hour operating window. The results of the numerical analysis are presented in Table 1. The size of the problem is noticed in the ''Instance'' column. EBB, BB, BBP, and IBM's OPL algorithms are placed in the ''Algorithm'' column. The value of the objective function (the profit as dollars) for every single solution is placed in the ''Objective'' column. Duration of solution as milliseconds is given in the ''Duration'' column. Finally, the total generated nodes during the search are given in the ''Num. of Branches'' column.

TABLE 1. Comparison of the algorithms.

Instance	Algorithm	Objective	Duration (msec)	Num. of Branches
24 Hr	EBB	14456.7936	5	72
	BB	14456.7936	5	72
	BBP	**14456.7936**	**4**	**69**
	OPL	14456.7936	350	96320
48 Hr	EBB	35487.6426	110	7088
	BB	35487.6426	35	1012
	BBP	**35487.6426**	**33**	**998**
	OPL	35487.6426	3680	743696
65 Hr	EBB	56403.1977	187	17681
	BB	56403.1977	102	6107
	BBP	**56403.1977**	**101**	**6065**
	OPL	56403.1977	26310	1408610
96 Hr	**EBB**	**75516.0752**	**454**	**41867**
	BB	75516.0752	1540	44370
	BBP	75516.0752	1407	41534
	OPL	75264.6704	68650	2793116
120 Hr	**EBB**	**87668.8694**	**2440**	**87147**
	BB	87668.8694	285540	417134
	BBP	87668.8694	270080	388933
	OPL	86016.7302	102350	6794606
240 Hr	**EBB**	**224577.3733**	**3124**	**131713**
17039Hr	**EBB**	**77176892.243**	**8 Hr**	**914 MM**

Results show that the EBB and the BB are the same for the small size of problems because the EBB requires two fluctuations (local optima) in the value of the objective function during the solving iterations. Otherwise, the EBB does not divide the problem and it behaves as the BB. In the context of small problems (48-hour and 65-hour instances), the BB and the BBP prove to be faster than the EBB. As the problem size increases, the EBB starts to find the solution much more quickly than the BB and BBP. In the case of small problems, the OPL is the solver that reaches the optimal solution most slowly, but as the problem size increases, the OPL also begins to find the solution faster than the BB and BBP, albeit deviating from optimality.

V. CONCLUSION

The study aimed to enhance the BB to facilitate the optimal solution of NP-Hard problems in the field of engineering. To achieve this goal, the approach involved partitioning the problem into independent partial problems in a manner that would not compromise optimality. Finally, we combined all solutions of the partial problems to obtain a global solution. For the partitioning of the problem, the objective function value needed to undergo two fluctuations, which means the problem must contain more than one local optimum for partitioning. The fluctuations in the value involve a decrease in maximization problems and an increase in minimization problems. When the second fluctuation ends, the problem can be partitioned from the point where the first fluctuation occurred. To substantiate this mathematically, the proposed hypothesis was applied to the energy dispatch optimization problem.

The optimal solution of the model presented in Equation 5-14 was found by the EBB in all experiments up to scenarios of 120 hours. However, in scenarios of 240 hours and 710 days, both the OPL, the BB, and the BBP failed to solve the problem. According to the obtained results, The EBB was capable of solving the same problem with significantly fewer nodes than the BB and BBP. The BBP generated approximately 10% fewer nodes than BB, whereas EBB generated approximately 80% fewer nodes than BB did. Nevertheless, in tests with a maximum solution time of 8 hours allocated to the algorithms, the EBB could only solve approximately 2 years of a 20-year dataset. In contrast, the BB, BBP, and the OPL could only find solutions for up to 5 days. Therefore, the first 5-day segment was compared, revealing that the EBB achieved solutions more rapidly and found optimal results as the problem size increased. The results demonstrate that EBB performs better than other algorithms as the problem size increases. However, to be certain, we will broaden the context of the study by applying the EBB to various problems in future research.

REFERENCES

[1] S. Zhang and S. Liu, ''A discrete improved artificial bee colony algorithm for 0–1 knapsack problem,'' *IEEE Access*, vol. 7, pp. 104982–104991, 2019, doi: 10.1109/ACCESS.2019.2930638.

[2] H. M. H. Saad, R. K. Chakrabortty, S. Elsayed, and M. J. Ryan, ''Quantum-inspired genetic algorithm for resource-constrained project-scheduling,'' *IEEE Access*, vol. 9, pp. 38488–38502, 2021, doi: 10.1109/ACCESS.2021.3062790.

[3] J.-R. Jiang and C.-W. Chu, ''Classifying and benchmarking quantum annealing algorithms based on quadratic unconstrained binary optimization for solving NP-hard problems,'' *IEEE Access*, vol. 11, pp. 104165–104178, 2023, doi: 10.1109/ACCESS.2023.3318206.

[4] C. P. Tomazella and M. S. Nagano, ''A comprehensive review of branch-and-bound algorithms: Guidelines and directions for further research on the flowshop scheduling problem,'' *Exp. Syst. Appl.*, vol. 158, Nov. 2020, Art. no. 113556, doi: 10.1016/j.eswa.2020.113556.

[5] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell, ''Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning,'' *Discrete Optim.*, vol. 19, pp. 79–102, Feb. 2016, doi: 10.1016/j.disopt.2016.01.005.

[6] L. Huang, X. Chen, W. Huo, J. Wang, F. Zhang, B. Bai, and L. Shi, ''Branch and bound in mixed integer linear programming problems: A survey of techniques and trends,'' 2021, *arXiv:2111.06257*.

[7] Y. Cui, Y. Yang, X. Cheng, and P. Song, "A recursive branch-and-bound algorithm for the rectangular guillotine strip packing problem," *Comput. Operations Res.*, vol. 35, no. 4, pp. 1281–1291, Apr. 2008, doi: 10.1016/j.cor.2006.08.011.
[8] Y. Cui and Y. Yang, "A recursive branch-and-bound algorithm for constrained homogenous T-shape cutting patterns," *Math. Comput. Model.*, vol. 54, nos. 5–6, pp. 1320–1333, Sep. 2011, doi: 10.1016/j.mcm.2011.04.003.
[9] V. Kariwala and Y. Cao, "Branch and bound method for multiobjective pairing selection," *Automatica*, vol. 46, no. 5, pp. 932–936, May 2010, doi: 10.1016/j.automatica.2010.02.014.
[10] E. A. Pruul, G. L. Nemhauser, and R. A. Rushmeier, "Branch-and-bound and parallel computation: A historical note," *Operations Res. Lett.*, vol. 7, no. 2, pp. 65–69, Apr. 1988, doi: 10.1016/0167-6377(88)90067-3.
[11] P. S. Laursen, "Simple approaches to parallel branch and bound," *Parallel Comput.*, vol. 19, no. 2, pp. 143–152, Feb. 1993, doi: 10.1016/0167-8191(93)90044-I.
[12] A. G. Mersha and S. Dempe, "Linear bilevel programming with upper level constraints depending on the lower level solution," *Appl. Math. Comput.*, vol. 180, no. 1, pp. 247–254, Sep. 2006, doi: 10.1016/j.amc.2005.11.134.
[13] F. Rendl, G. Rinaldi, and A. Wiegele, "A branch and bound algorithm for max-cut based on combining semidefinite and polyhedral relaxations," in *Integer Programming and Combinatorial Optimization* (Lecture Notes in Computer Science: Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 4513. Berlin, Germany: Springer, 2007, doi: 10.1007/978-3-540-72792-7_23.
[14] A. Przybylski and X. Gandibleux, "Multi-objective branch and bound," *Eur. J. Oper. Res.*, vol. 260, no. 3, pp. 856–872, Aug. 2017, doi: 10.1016/j.ejor.2017.01.032.
[15] A. J. Woodcock and J. M. Wilson, "A hybrid Tabu search/branch & bound approach to solving the generalized assignment problem," *Eur. J. Oper. Res.*, vol. 207, no. 2, pp. 566–578, Dec. 2010, doi: 10.1016/j.ejor.2010.05.007.
[16] G. G. Zabudsky and N. S. Veremchuk, "Branch and bound method for the weber problem with rectangular facilities on lines in the presence of forbidden gaps," in *Communications in Computer and Information Science*, vol. 871. Cham, Switzerland: Springer, 2018, doi: 10.1007/978-3-319-93800-4_3.
[17] N. Adelgren and A. Gupte, "Branch-and-bound for biobjective mixed-integer programming," 2017, *arXiv:1709.03668*.
[18] W. M. Elaibi, "Branch and bound algorithm and an improvement for calculating the nearest link of building a railway network," *Al-Mustansiriyah J. Sci.*, vol. 31, no. 4, pp. 114–119, Dec. 2020, doi: 10.23851/mjs.v31i4.923.
[19] R. Alvarez-Valdes, A. Martinez, and J. M. Tamarit, "A branch & bound algorithm for cutting and packing irregularly shaped pieces," *Int. J. Prod. Econ.*, vol. 145, no. 2, pp. 463–477, Oct. 2013, doi: 10.1016/j.ijpe.2013.04.007.
[20] C. R. N. Estevam, M. J. Rider, E. Amorim, and J. R. S. Mantovani, "Reactive power dispatch and planning using a non-linear branch-and-bound algorithm," *IET Gener., Transmiss. Distribution*, vol. 4, no. 8, p. 963, 2010, doi: 10.1049/iet-gtd.2009.0422.
[21] I. Lendák, A. Vidács, and A. Erdeljan, "Electric power system one-line diagram generation with branch and bound algorithm," in *Proc. IEEE Int. Energy Conf. Exhib. (ENERGYCON)*, Sep. 2012, pp. 947–951, doi: 10.1109/ENERGYCON.2012.6348286.
[22] S. G. Constante F., J. C. López, and M. J. Rider, "Optimal reactive power dispatch with discrete controllers using a branch-and-bound algorithm: A semidefinite relaxation approach," *IEEE Trans. Power Syst.*, vol. 36, no. 5, pp. 4539–4550, Sep. 2021, doi: 10.1109/TPWRS.2021.3056637.
[23] A. Farag, S. Al-Baiyat, and T. C. Cheng, "Economic load dispatch multiobjective optimization procedures using linear programming techniques," *IEEE Trans. Power Syst.*, vol. 10, no. 2, pp. 731–738, May 1995, doi: 10.1109/59.387910.
[24] J. Sharples. (2023). *Ontario Energy Demand*. [Online]. Available: https://www.kaggle.com/datasets/jacobsharples/ontario-electricity-demand
[25] M. Goerigk, B. Grün, and P. Heßler, "Branch and bound algorithms for the bus evacuation problem," *Comput. Operations Res.*, vol. 40, no. 12, pp. 3010–3020, Dec. 2013, doi: 10.1016/j.cor.2013.07.006.
[26] S. Coniglio, F. Furini, and P. San Segundo, "A new combinatorial branch-and-bound algorithm for the knapsack problem with conflicts," *Eur. J. Oper. Res.*, vol. 289, no. 2, pp. 435–455, Mar. 2021, doi: 10.1016/j.ejor.2020.07.023.
[27] P. San Segundo, F. Furini, and J. Artieda, "A new branch-and-bound algorithm for the maximum weighted clique problem," *Comput. Operations Res.*, vol. 110, pp. 18–33, Oct. 2019, doi: 10.1016/j.cor.2019.05.017.

ELIF CESUR received the dual bachelor's degree in industrial and computer engineering from İstanbul Kültür University, in 2008, and the Ph.D. degree from the Department of Mechanical Engineering, Technical University Dortmund, Dortmund, Germany, in 2015. She is currently an Assistant Professor with the Department of Industrial Engineering, Istanbul Medeniyet University, İstanbul, Türkiye. She is also a Postdoctoral Researcher with Machine Intelligence Research Laboratories (MIR Labs). Additionally, she is involved in several significant industrial projects related to priority areas for Türkiye: 1) digitalization of locomotives: anomaly detection and predictive maintenance via digital twin of the locomotive of TÜLOMSAŞ (2020-Supervisor); 2) capacity and layout planning of Turkish Airlines Cargo Terminal (2018-Developer); and 3) logistics on demand: usage-dependent simulation for anticipative change planning of intra-logistic systems, DFG Germany (2015-Developer).

MUHAMMET RAŞIT CESUR received the Ph.D. degree from the Department of Industrial Engineering, Sakarya University, Sakarya, Türkiye, in 2019. He is currently an Assistant Professor with the Department of Industrial Engineering, Istanbul Medeniyet University, İstanbul, Türkiye. Additionally, he is the Co-Founder of ARVASIS Information and Consultancy Inc. He has been involved in various projects, including the development of a digital twin system for anomaly detection and predictive maintenance in locomotives, in 2020, the design of the 2D IoT intelligent camera, from August 2020 to December 2021, the design of an in-line quality control system for PVC profile production with a focus on outlier detection in product images for quality control for Deceuninck, in 2019, and research on artificial intelligence for product identification in the shipment process for Toyota-Boshoku, in 2017.

AJITH ABRAHAM (Senior Member, IEEE) received the B.Tech. degree in electrical and electronic engineering from the University of Calicut, in 1990, the M.S. degree from Nanyang Technological University, Singapore, in 1998, and the Ph.D. degree in computer science from Monash University, Melbourne, Australia, in 2001. He is currently the Pro-Vice Chancellor of Bennett University, India. Prior to this, he was the Dean of the Faculty of Computing and Mathematical Sciences, FLAME University, Pune, and the Founding Director of Machine Intelligence Research Laboratories (MIR Labs), USA, a not-for-profit scientific network for innovation and research excellence, connecting industry and academia. During the last three years, he also held two university professorial appointments, including a Professor of artificial intelligence with Innopolis University, Russia, and the Yayasan Tun Ismail Mohamed Ali Professorial Chair of Artificial Intelligence, UCSI, Malaysia. He is involved on a multi-disciplinary environment and has authored/co-authored more than 1400 research publications. He has more than 54 000 academic citations (H-index is more than 110 as per Google Scholar). He has given more than 200 plenary lectures and conference tutorials (in more than 20 countries). He was the Chair of the IEEE Systems, Man and Cybernetics Society Technical Committee on Soft Computing (which has over more than 200 members), from 2008 to 2021. He was the Editor-in-Chief of *Engineering Applications of Artificial Intelligence* (EAAI), from 2016 to 2021. He serves/served on the editorial board for over 15 international journals indexed by Thomson ISI. He served as a Distinguished Lecturer for the IEEE Computer Society Representing Europe, from 2011 to 2013.